

Using Automatic Differentiation for Second-Order Matrix-free Methods in PDE-constrained Optimization

P. D. Hovland* D. E. Keyes† L. C. McInnes‡ W. Samyono§

Abstract

Classical methods of constrained optimization are often based on the assumptions that projection onto the constraint manifold is routine but accessing second-derivative information is not. Both assumptions need revision for the application of optimization to systems constrained by partial differential equations, in the contemporary limit of millions of state variables and in the parallel setting. Large-scale PDE solvers are complex pieces of software that exploit detailed knowledge of architecture and application and cannot easily be modified to fit the interface requirements of a blackbox optimizer. Furthermore, in view of the expense of PDE analyses, optimization methods not using second derivatives may require too many iterations to be practical. For general problems, automatic differentiation is likely to be the most convenient means of exploiting second derivatives. We delineate a role for automatic differentiation in matrix-free optimization formulations involving Newton's method, in which little more storage is required than that for the analysis code alone.

1 Introduction

Years of two-sided (from architecture up, from applications down) algorithms research has made it possible to solve partial differential equation (PDE) problems implicitly with reasonable scalability. PDEs are equality constraints on the state variables in many optimization problems. Hardly auxiliary, the PDE system may contain millions of degrees of freedom. In problems of shape optimization and control, the number of optimization parameters is typically much smaller than the number of state variables. In problems of parameter identification, the number of parameters to be optimized may be comparable to the number of state variables, but few general-purpose optimization frameworks have been demonstrated at the scale required for three-dimensional problems. We therefore propose that large-scale PDE-constrained optimization codes usually should be constructed around the data structures and functional capabilities of the PDE solver.

Optimization is easily incorporated through the Lagrange saddle-point formulation into a Newton-like parallel PDE framework that accommodates substructuring. Newton's method is a common element in the most rapidly convergent solvers and optimizers. Furthermore, a PDE solver that is not part of an optimization framework is probably short of what the client really wants. Hence, for both algorithmic and teleological reasons, analysis and optimization belong together.

We focus in Section 2 on the Newton-Krylov-Schwarz (NKS) family of parallel implicit rootfinders, and we give an example of pseudo-transient globalization of NKS (Ψ NKS) in a large-scale parallel context, aerodynamics. The first-order optimality conditions of equality-constrained optimization using the Lagrangian are presented in Section 3, which introduces a parallel optimization framework called

*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844
<http://www.mcs.anl.gov/~hovland>

†Mathematics & Statistics Department, Old Dominion University, Norfolk, VA 23529-0077; ISCR, Lawrence Livermore National Laboratory, Livermore, CA 94551-0808; and ICASE, NASA Langley Research Center, Hampton, VA 23681-2199
<http://www.math.odu.edu/~keyes>

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844
<http://www.mcs.anl.gov/~mcinnes>

§Mathematics & Statistics Department, Old Dominion University, Norfolk, VA 23529-0077

LNKS (Lagrange-Newton-Krylov-Schur or Lagrange-Newton-Krylov-Schwarz). In Section 4 we sketch a prototype parameter identification example from the field of radiation transport. The complexity of LNKS when automatic differentiation (AD) is employed in the Krylov matrix-vector operation is discussed in Section 5. Finally, in Section 6 we summarize our work and indicate some future directions.

2 Newton-Krylov-Schwarz

In this section, we describe the NKS framework from the inside outward, then illustrate it in a large-scale parallel context.

2.1 Schwarz

Schwarz [10, 13, 22] methods are solvers or preconditioners that create concurrency at a desired granularity algorithmically and explicitly through partitioning, without the necessity of any code dependence analysis or special compiler. Generically, in continuous or discrete settings, Schwarz partitions a solution space into n subspaces, possibly overlapping, whose union is the original space, and forms an approximate inverse of the operator in each subspace. Algebraically, to solve the discrete linear system, $Ax = f$, let Boolean rectangular matrix R_i extract the i^{th} subset of the elements of x : $x_i = R_i x$, and let $A_i = R_i A R_i^T$. Then the Schwarz approximate inverse, B^{-1} , is defined as $\sum_i R_i^T A_i^{-1} R_i$. From the PDE perspective, subspace decomposition is domain decomposition. We form $B^{-1} \approx A^{-1}$ out of (approximate) local solves on (possibly overlapping) subdomains, as in Fig. 1. This can be used to iterate in a stationary way, as a splitting matrix: $x^{k+1} = (I - B^{-1}A)x^k + B^{-1}f$. However, since $\rho(I - B^{-1}A)$ may be greater than unity in general, this additive splitting may not converge as a stationary iteration. “Multiplicative” Schwarz methods (Gauss-Seidel-like, relative to the Jacobi-like “additive” above) can be proved convergent when A derives from an elliptic PDE, under certain partitionings.

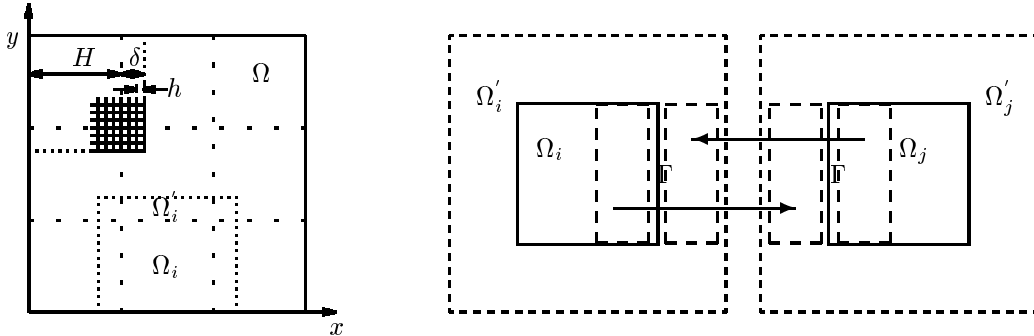


Figure 1: **Left:** A domain Ω partitioned into nine overlapping subdomains, Ω_i , extended slightly by overlapping to subdomains Ω'_i , showing the scales of the mesh spacing (h), the subdomain overlap (δ), and the subdomain diameter (H). **Right:** Two adjacent subdomains with common edge Γ pulled apart to show overlap regions as separate buffers, which are implemented in the local data structures of each.

In the PDE context, Boolean operators R_i and R_i^T , $i = 1, \dots, n$, represent gather and scatter (communication) operations, mapping between a global vector and its i^{th} subdomain support. When A derives from an elliptic operator and R_i is the characteristic function of unknowns in a subdomain, optimal convergence (independent of $\dim(x)$ and the number of partitions) can be proved, with the addition of a coarse grid, which is denoted with subscript “0”: $B^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i>0} R_i^T A_i^{-1} R_i$. Here, R_0 is a conventional geometrically based multilevel interpolation operator. It is an important freedom in practical implementations that the coarse grid space need not be related to the fine grid space or to the subdomain partitioning.

The A_i^{-1} ($i > 0$) in B^{-1} are often replaced with inexact solves in practice, denoted by \tilde{A}_i^{-1} . The exact forward matrix-vector action of A in $B^{-1}A$ is still required, even if inexact solves are employed in the preconditioner.

Condition number estimates for $B^{-1}A$ are given in Table 1 for generous overlap $\delta = \mathcal{O}(H)$. Otherwise, if $\delta \ll H$, the two-level result is $\mathcal{O}(1 + H/\delta)$. The two-level Schwarz method with generous overlap has a condition number that is independent of the fineness of the discretization and the granularity of the decomposition, which implies perfect algorithmic scalability. However, there is an increasing implementation overhead in the coarse-grid solution required in the two-level method that offsets this perfect algorithmic scalability. In practice, a one-level method is often used, since it is amenable to a perfectly scalable implementation. These condition number results are extensible to nonself-adjointness, mild indefiniteness, and inexact subdomain solvers. The theory requires a “sufficiently fine” coarse mesh, H , for the first two of these extensions, but computational experience shows that the theory is often pessimistic.

2.2 Krylov-Schwarz

Although the spectral radius, $\rho(I - B^{-1}A)$, may exceed unity, the spectrum, $\sigma(B^{-1}A)$, is profoundly clustered, so Krylov acceleration methods should work well on the preconditioned solution of $B^{-1}Ax = B^{-1}f$. Krylov-Schwarz methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system. For convergence scalability estimates, assume one subdomain per processor in a d -dimensional isotropic problem, where $N = h^{-d}$ and $P = H^{-d}$. Then iteration counts may be estimated as in the last two columns of Table 1.

Table 1: Theoretical condition number estimates $\kappa(B^{-1}A)$, for self-adjoint positive-definite elliptic problems [22] and corresponding iteration count estimates for Krylov-Schwarz based on an idealized isotropic partitioning of the domain in two or three dimensions.

<i>Preconditioning</i>	$\kappa(B^{-1}A)$	<i>2D Iter.</i>	<i>3D Iter.</i>
Point Jacobi	$\mathcal{O}(h^{-2})$	$\mathcal{O}(N^{1/2})$	$\mathcal{O}(N^{1/3})$
Domain Jacobi	$\mathcal{O}((hH)^{-1})$	$\mathcal{O}((NP)^{1/4})$	$\mathcal{O}((NP)^{1/6})$
1-level Additive Schwarz	$\mathcal{O}(H^{-2})$	$\mathcal{O}(P^{1/2})$	$\mathcal{O}(P^{1/3})$
2-level Additive Schwarz	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

2.3 Newton-Krylov-Schwarz

Let $F(x) = 0$ be a discrete system of nonlinear equations arising from an elliptically dominated system of PDEs. Let its Jacobian be denoted $J \equiv \partial F/\partial x$. Inexact Newton iteration on $F(x) = 0$, involves selecting an initial iterate $x^{(0)}$ and iterating for a correction to the current $x^{(k)}$: $x^{(k+1)} = x^{(k)} + \lambda_k \delta x$, where $\|J(x^{(k)})\delta x + F(x^{(k)})\| < \eta_k$. A large body of literature exists on how to choose η_k and λ_k for robustness and efficiency. Any of these members of the inexact Newton family of algorithms may be implemented as a Newton-Krylov-Schwarz method, by iterating for δx with a linear Krylov-Schwarz method. Partitioning x induces block structure on the Jacobian matrix. As anticipated in the presentation of Schwarz above, we do not need any Jacobians explicitly; rather, matrix-vector action of the Jacobian at point $x^{(k)}$ may be performed with finite Fréchet differencing (FD) or automatic differentiation (AD) about the point, and preconditioning of the Jacobian is done with approximate local operators, approximately solved in accordance with overall performance trade-offs.

Newton-Krylov-Schwarz has been demonstrated to be an effective parallel implicit solver for large-scale nonlinear problems derived from PDEs (see, e.g., P. Brown and collaborators at LLNL [7, 8] and D. Knoll and collaborators at LANL [17, 19]). It has been applied to problems in aerodynamics, radiation transport, porous media, semiconductors, geophysics, astrophysical MHD, population dynamics, and other fields. It has been implemented in a parallel matrix-free object-oriented framework, including both FD and AD distributed matvecs, in PETSc software from Argonne [2].

We advocate employing Ψ NKS in a split-discretization formulation, in which economizations are taken in the left-hand side preconditioner blocks of J relative to the more accurate, physical discretization-dictated right-hand operator for J . Examples of such economizations include sacrificed coupling for pro-

cess concurrency, segregation of physics into successive phases with simple structure (operator-splitting), the Jacobian of a lower-order discretization for fewer nonzeros and fewer colors in a minimal coloring, the Jacobian of a related discretization allowing “fast” solves, a Jacobian with lagged values for any terms that are expensive to compute or small or both, and a Jacobian stored in half precision for superior (nearly doubled) memory bandwidth, as measured in words per second, in the bandwidth-limited linear algebra routines of a sparse, unstructured PDE solver.

2.4 Pseudo-Transient Newton-Krylov-Schwarz

NKS is commonly robustified with pseudo-transience (Ψ NKS) [15, 21] or other continuation strategies. In Ψ NKS one solves $F(x) = 0$ through a series of modified problems

$$H_\ell(x) \equiv \frac{x - x_{\ell-1}}{\delta t_\ell} + F(x) = 0, \quad \ell = 1, 2, \dots,$$

each of which is solved (approximately) for x_ℓ . This sequence hugs a physical transient when δt_ℓ is small, for which the associated diagonally dominant Jacobians are well conditioned. δt_ℓ is advanced from $\delta t_0 \ll 1$ to $\delta t_\ell \rightarrow \infty$ as $\ell \rightarrow \infty$, so that x_ℓ approaches the root of $F(x) = 0$. Unlike many robustification techniques, Ψ NKS does *not* require reduction in $\|F(x)\|$ at each step; its ability to climb hills in the residual norm is useful in problems with complex physics, such as combustion, in which a local minimum (e.g., extinction) may not be the physically desired one.

2.5 Example from Computational Aerodynamics

To illustrate the effectiveness of NKS in practice, we quote below some performance data for a computational aerodynamics problem, which won a 1999 Gordon Bell prize [1]. The Euler equations were solved on a tetrahedral unstructured grid for the flow over an ONERA M6 wing.

The finest-granularity decomposition consisted of 3072 subdomains on a grid of approximately 2.8M vertices. Each subdomain was computed on a pair of Intel Pentium Pro processors (6144 processors altogether) on the ASCI Red machine at Sandia, which executed in shared-memory OpenMP mode on the evaluation of $F(x)$, while the linear algebra portions of the computation were left single-threaded on each node. Up to 0.227 Tflop/s were achieved; see [1, 14] for details.

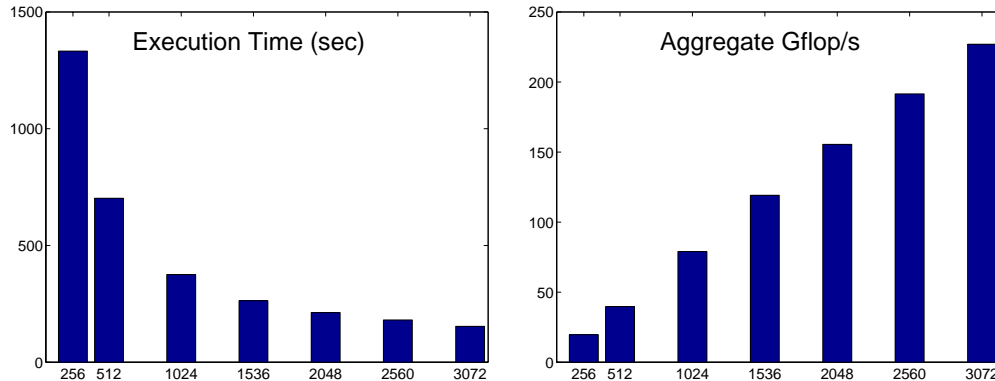


Figure 2: Execution time and aggregate flop rate for Ψ NKS on incompressible Euler flow over an ONERA M6 wing, on a tetrahedral grid of 2,761,744 vertices, based on the KMeTiS-PETSc implementation of the NASA code FUN3D run on up to 3072 nodes of ASCI Red.

3 Implications of NKS for Optimization

Equality constrained optimization leads, through the Lagrangian formulation, to a multivariate non-linear rootfinding problem for the gradient (the first-order necessary conditions), which is amenable to treatment by Newton’s method. To establish notation, consider the following canonical framework, in which we enforce equality constraints on the state variables only. (Design variable constraints require additional notation, and inequality constraints require additional algorithmics, but these generalizations are well understood.) Choose m design variables u to minimize the objective function, $\phi(u, x)$, subject to n state constraints, $h(u, x) = 0$, where x is the vector of state variables. In the Lagrange framework, a stationary point of the Lagrangian function

$$\mathcal{L}(x, u, \lambda) \equiv \phi(x, u) + \lambda^T h(x, u)$$

is sought. When Newton’s method is applied to the first-order optimality conditions, a linear system known as the Karush-Kuhn-Tucker (KKT) system arises at each step. There is a natural “outer” partitioning: the vector of parameters is often of lower dimension than the vectors of states and multipliers. This suggests a Schur complement-like block elimination process at the outer level, not for concurrency, but for numerical robustness and conceptual clarity. Within the state-variable subproblem, which must be solved repeatedly in the Schur complement reduction, Schwarz provides a natural “inner” partitioning for concurrency.

A major choice to be made in the Newton approach to constrained optimization is between exact elimination of the states and multipliers by satisfying constraint feasibility at every step (reduced system), and progress in all variables simultaneously, possibly violating constraints on intermediate iterates (full system). An advantage of the former is the existence of high-quality, robust blackbox software for this so-called reduced sequential quadratic programming (RSQP) approach. The advantages of the latter are in reuse of high-quality parallel PDE software, the freedom to use inexact solves (since finely resolved PDE discretizations in 3D militate against exact elimination), and the ease of application of automatic differentiation software, without having to differentiate through the nonlinear subiterations that would be implied by repeated projection to the constraint manifold in RSQP.

We mention three classes of PDE-constrained optimization:

- **Design optimization** (especially shape optimization): u parameterizes the domain of the PDE (e.g., a lifting surface) and ϕ is a cost-to-benefit ratio of forces, energy expenditures, and so forth. Typically, m is small compared with n and does not scale directly with it. However, m may still be several hundred in industrial applications.
- **Optimal control**: u parameterizes a continuous control function acting on the surface of the domain, and ϕ is the norm of the difference between desired and actual responses of the system. Typically, $m \propto n^{2/3}$.
- **Parameter identification/data assimilation**: u parameterizes an unknown continuous constitutive or forcing function defined throughout the domain, and ϕ is the norm of the difference between measurements and simulation results. Typically, $m \propto n$.

Written out in partial detail, the optimality conditions are

$$\frac{\partial \mathcal{L}}{\partial x} \equiv \frac{\partial \phi}{\partial x} + \lambda^T \frac{\partial h}{\partial x} = 0, \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial u} \equiv \frac{\partial \phi}{\partial u} + \lambda^T \frac{\partial h}{\partial u} = 0, \quad (2)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} \equiv h = 0. \quad (3)$$

Newton’s method iteratively seeks a correction,

$$\begin{pmatrix} \delta x \\ \delta u \\ \delta \lambda \end{pmatrix} \quad \text{to the iterate} \quad \begin{pmatrix} x \\ u \\ \lambda \end{pmatrix}.$$

With subscript notation for partial derivatives, the Newton correction (KKT) equations are

$$\begin{bmatrix} (\phi_{,xx} + \lambda^T h_{,xx}) & (\phi_{,xu} + \lambda^T h_{,xu}) & h_{,x}^T \\ (\phi_{,ux} + \lambda^T h_{,ux}) & (\phi_{,uu} + \lambda^T h_{,uu}) & h_{,u}^T \\ h_{,x} & h_{,u} & 0 \end{bmatrix} \begin{pmatrix} \delta x \\ \delta u \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \phi_{,x} + \lambda^T h_{,x} \\ \phi_{,u} + \lambda^T h_{,u} \\ h \end{pmatrix}$$

or

$$\begin{bmatrix} W_{xx} & W_{ux}^T & J_x^T \\ W_{ux} & W_{uu} & J_u^T \\ J_x & J_u & 0 \end{bmatrix} \begin{pmatrix} \delta x \\ \delta u \\ \lambda_+ \end{pmatrix} = - \begin{pmatrix} g_x \\ g_u \\ h \end{pmatrix}, \quad (4)$$

where $W_{ab} \equiv \frac{\partial^2 \phi}{\partial a \partial b} + \lambda^T \frac{\partial^2 h}{\partial a \partial b}$, $J_a \equiv \frac{\partial h}{\partial a}$, and $g_a = \frac{\partial \phi}{\partial a}$, for $a, b \in \{x, u\}$, and where $\lambda_+ = \lambda + \delta \lambda$.

3.1 Newton Reduced SQP

The RSQP method [20] consists of a three-stage iteration. We follow the language and practice of [4, 5] in this and the next subsection.

- **Design Step** (Schur complement for middle blockrow):

$$H \delta u = f,$$

where H and f are the reduced Hessian and gradient, respectively:

$$\begin{aligned} H &\equiv W_{uu} - J_u^T J_x^{-T} W_{ux}^T + (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} J_u \\ f &\equiv -g_u + J_u^T J_x^{-T} g_x - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} h \end{aligned}$$

- **State Step** (last blockrow):

$$J_x \delta x = -h - J_u \delta u$$

- **Adjoint Step** (first blockrow):

$$J_x^T \lambda_+ = -g_x - W_{xx} \delta x - W_{ux}^T \delta u$$

In each overall iteration, we must form and solve with the reduced Hessian matrix H , and we must solve separately with J_x and J_x^T . The latter two solves are almost negligible compared with the cost of forming H , which is dominated by the cost of forming the sensitivity matrix $J_x^{-1} J_u$. Because of the quadratic convergence of Newton, the number of overall iterations is few (asymptotically independent of m). However, the cost of forming H at each design iteration is m solutions with J_x . These are potentially concurrent over independent columns of J_u , but prohibitive.

In order to avoid computing any Hessian blocks, the design step may be approached in a quasi-Newton (e.g., BFGS) manner [20]. Hessian terms are dropped from the adjoint step RHS.

- **Design Step** (severe approximation to middle blockrow):

$$Q \delta u = -g_u + J_u^T J_x^{-T} g_x,$$

where Q is a quasi-Newton approximation to the reduced Hessian

- **State Step** (last blockrow):

$$J_x \delta x = -h - J_u \delta u$$

- **Adjoint Step** (approximate first blockrow):

$$J_x^T \lambda_+ = -g_x$$

In each overall iteration of quasi-Newton RSQP, we must perform a low-rank update on Q or its inverse, and we must solve with J_x and J_x^T . This strategy vastly reduces the cost of an iteration; however, it is no longer a Newton method. The number of overall iterations is many. Since BFGS is equivalent to unpreconditioned CG for quadratic objective functions, $\mathcal{O}(m^p)$ sequential cycles ($p > 0$, $p \approx \frac{1}{2}$) may be anticipated. Hence, quasi-Newton RSQP is not scalable in the number of design variables, and no ready form of parallelism can address this convergence-related defect.

To summarize, conventional RSQP methods apply a (quasi-)Newton method to the optimality conditions: solving an approximate $m \times m$ system to update u , updating x and λ consistently (to eliminate them), and iterating. The unpalatable expense arises from the exact linearized analyses for updates to x and λ that appear in the inner loop. We therefore consider replacing the exact elimination steps of RSQP with preconditioning steps in an outer loop, as described in the next subsection.

3.2 Full Space Lagrange-NKS Method

The new philosophy is to apply a Krylov-Schwarz method directly to the $(2n + m) \times (2n + m)$ KKT system (4). For this purpose, we require the action of the full matrix on the full-space vector and a good full-system preconditioner, for algorithmic scalability. One Newton SQP iteration is a perfect preconditioner—a block factored solver, based on forming the reduced Hessian of the Lagrangian H —but, of course, far too expensive. Backing off wherever storage or computational expense becomes impractical for large-scale PDEs generates a family of attractive methods.

To precondition the full system, we need approximate inverses to the three left-hand side matrices in the first algorithm of Section 3.1, namely, H , J , and J^T . If a preconditioner is available for H , and exact solves are available for J , and J^T , then it may be shown [16] that conjugate gradient Krylov iteration on the (assumed symmetrizable) reduced system and conjugate gradient iteration on the full system yield the same sequence of iterates. The iterates are identical in the sense that if one were to use the values of u arising from the iteration on the reduced system in the right-hand side of the block rows for x and λ , one would reconstruct the iterates of the full system, when the same preconditioner used for H in the reduced system is used for the W_{uu} block in the full system. Moreover, the spectrum of the full system is simply the spectrum of the reduced system supplemented with a large multiplicity of unit eigenvalues. If one retreats from exact solves with J and J^T , the equivalence no longer holds; however, if good preconditioners are used for these Jacobian blocks, then the cloud of eigenvalues around unity is still readily shepherded by a Krylov method, and convergence should be nearly as rapid as in the case of exact solves.

This Schur-complement-based preconditioning of the full system by was proposed in this equality-constrained optimization context by Biros and Ghattas in 1998 [4] and earlier in a closely related context by Batterman and Heinkenschloss [3]. From a purely algebraic point of view, the same Schur-complement-based preconditioning was advocated by Keyes and Gropp in 1987 [16] in the context of domain decomposition. There, the reduced system was a set of unknowns on the interface between subdomains, and the savings from the approximate solves on the subdomain interiors more than paid for the modest degradation in convergence rate relative to interface iteration on the Schur complement. The main advantage of the full system problem is that the Schur complement never needs to be formed. Its exact action is felt on the design variable block through the operations carried out on the full system.

Biros and Ghattas have demonstrated the large-scale parallel effectiveness of the full system algorithm on a 3D Navier-Stokes flow boundary control problem, where the objective is dissipation minimization of flow over a cylinder using suction and blowing over the back portion of the cylinder as the control variables [5]. They performed this optimization with domain-decomposed parallelism on 128 processors of a T3E, using an original optimization toolkit add-on to the PETSc [2] toolkit. To quote one result from [5], for 6×10^5 state constraints and 9×10^3 controls, full-space LNKS with approximate subdomain solves beat quasi-Newton RSQP by an order of magnitude (4.1 hours versus 53.1 hours).

Two names have evolved for the new algorithm: Lagrange-Newton-Krylov-Schwarz was proposed by Keyes in May 1999 at the SIAM Conference on Optimization, and Lagrange-Newton-Krylov-Schur by Biros and Ghattas in [5]. The former emphasizes the use of NKS to precondition the large Jacobian blocks, the latter the use of Schur complements to precondition the overall KKT matrix. Both

preconditioner suffixes are appropriate in a nested fashion, so we propose Lagrange-Newton-Krylov-Schur-Schwarz (LNKSS) when both preconditioners are used (see Fig. 3).

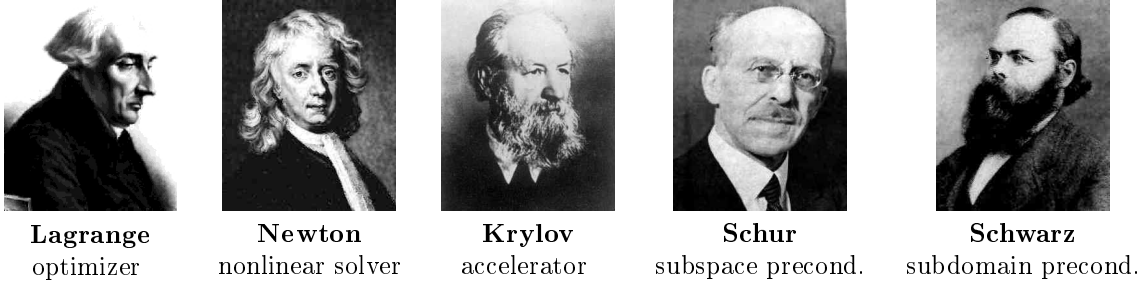


Figure 3: LNKSS: A Parallel Optimizer for BVP-constrained Problems

Automatic differentiation has two roles in the new algorithm: formation of the action on a Krylov vector of the full KKT matrix, including the full second-order Hessian blocks, and supply of approximations to the elements of J (and J^T) for the preconditioner. While the synergism of AD with LNKSS is in many ways obvious, advocacy of this novel combination is the primary thrust of the current paper.

4 Example of LNKS Parameter Identification

The effectiveness of Schwarz preconditioning is illustrated in the analysis context in Section 2.5. In this section, we illustrate Schur preconditioning and automatic differentiation in the parameter identification context. Schwarz and Schur techniques will be combined in a large-scale example from multidimensional radiation transport in the future. The governing constraint for our one-dimensional problem is the steady state version of the following radiation diffusion equation for material temperature:

$$\frac{\partial T}{\partial t} = \nabla \cdot (\beta(x) T^\alpha \nabla T). \quad (5)$$

Instead of solving the impulsive Marshak wave formulation of this problem, we ignore the time derivative and impose Dirichlet boundary conditions of 1.0 and 0.1, respectively, on $T(x)$ at the left- and right-hand endpoints of the unit interval. The resulting ODE boundary value problem is discretized with centered finite differences. The state variables are the discrete temperatures at the mesh nodes, and the design variables are the parameters α and $\beta(x)$. The cost function is temperature matching, $\phi(u, T) = \frac{1}{2} \|T(x) - \bar{T}(x)\|^2$, where \bar{T} is based on a given α , $\beta(x)$ profile. These parameters are specified for the computation of $\bar{T}(x)$, and then “withheld,” to be determined by the optimizer. More generally, $\bar{T}(x)$ would be a desired or experimentally measured profile, and the phenomenological law and material specification represented by α and $\beta(x)$ would be determined to fit. The Brisk-Spitzer form of the nonlinear dependence of the diffusivity on the temperature is $\alpha = 2.5$. For \bar{T} we assume a jump in material properties at the midpoint of the interval: $\beta(x) = 1, 0 \leq x \leq \frac{1}{2}$ and $\beta(x) = 10, \frac{1}{2} < x \leq 1$.

Our initial implementation of LNKS is in the software framework of MATLAB [18] and ADMAT [12]. ADMAT is an automatic differentiation framework for MATLAB, based on operator overloading. After an `m`-file is supplied for the cost function and constraint functions, all gradients, Jacobians, and Hessians (as well as their transposes and their contracted action on vectors) used anywhere in the LNKS algorithm are computed automatically without further user effort. There is one exception in the current code: our almost trivial cost function (with no parametric dependence and separable quadratic state dependence) is differentiated by hand, yielding an identity matrix for ϕ_{xx} . Our preconditioner is the RSQP block factorization, except that the reduced Hessian preconditioner is replaced with the identity. The reduced Hessian preconditioner block should be replaced with a quasi-Newton reduced Hessian in the future. In the present simple experiments, Newton’s method is used without robustification of any kind. Figure 4 shows how the optimizer eventually finds the values of 2.5 for α and 10 for $\beta_{right} \equiv \beta(x)$

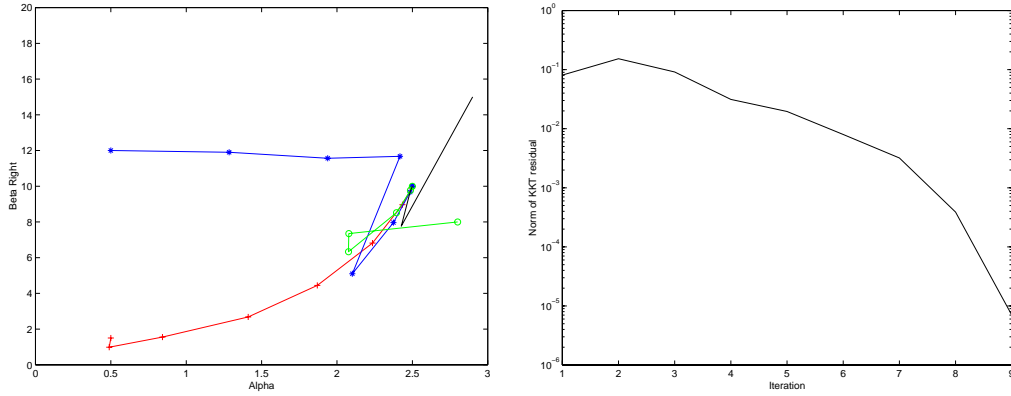


Figure 4: **Left:** Four convergence histories for (α, β_{right}) merged into one plot. **Right:** KKT norm convergence history for the parameter identification problem with initial iterate based on $\alpha = 0.5$, $\beta_{right} = 1.5$.

in the interval $\frac{1}{2} \leq x < 1$, from four different starting points of $(0.5, 1.5)$, $(0.5, 12)$, $(2.8, 8)$ and $(2.9, 15)$. A sample convergence history for the norm of the residual of (1)–(3) shows quadratic behavior.

Shown in Fig. 5 are initial and final distributions of $T(x)$ for α and β_{right} displaced as in Fig. 4 in all directions away from the “true” values of $(2.5, 10)$. The left-hand graphs show the “true” temperature profile to be matched (the curve common to all four plots) and the equilibrium temperature profile at the initial values of the parameters. Within each half-interval, the temperature gradient is sharper on the right (smaller values of $T(x)$), since the heat flux across every station is the same and the temperature-dependent diffusion coefficient factor inside the divergence operator of (5) is smaller in zones of smaller temperature. In regions with a larger β -factor, the overall average temperature drop is smaller by the same reasoning. Small α suppresses the nonlinear dependence of the diffusivity on temperature, so the initial temperature profiles are nearly linear within each constant- β region in the first two plots. Only in the first case is the approach to the true (α, β_{right}) monotonic, but plain Newton is robust enough to converge from all quadrants.

5 Complexity of Automatic Differentiation-based LNKs

Although our demonstration example is low dimensional, LNKs will generally be applied to large problems of n state variables and m parameters. Upon surveying existing AD tools, we conclude that the preconditioned matrix-vector product can be formed in time linear in these two parameters. The shopping list of matrix actions in forming the preconditioned Jacobian-vector product of LNKs is $W_{xx}, W_{uu}, W_{ux}, W_{ux}^T, J_u, J_u^T, J_x^{-1}, J_x^{-T}$, and H^{-1} .

The first six are needed in the full-system matrix-vector multiplication. For this multiplication we require “working accuracy” comparable to the state of the art in numerical differentiation.

Accurate action of the last three is required in RSQP but not in the full system preconditioner. We recommend approximate factorizations of lower-quality approximations, including possibly just W_{uu} for H , or a traditional quasi-Newton rank-updated approximation to the inverse.

We estimate the complexity of applying each block of the KKT Jacobian, assuming only that $h(x, u)$ is available in subroutine call form and that all differentiated blocks are from AD tools, such as the ADIC [6] tool we are using in a parallel implementation of LNKSS. We assume that J_x is needed, element by element, in order to factor it; hence, J_x^T is also available. Since these are just preconditioner blocks, we generally derive these elements from a different (cheaper) function call for the gradient of the Lagrangian than that used for the matvec. Define C_h , the cost of evaluating h ; p_x , 1 + the chromatic number of $J_x \equiv h_{,x}$; and p_u , 1 + the chromatic number of $J_u \equiv h_{,u}$. Then the costs of the Jacobian objects are shown in the first three rows of Table 2.

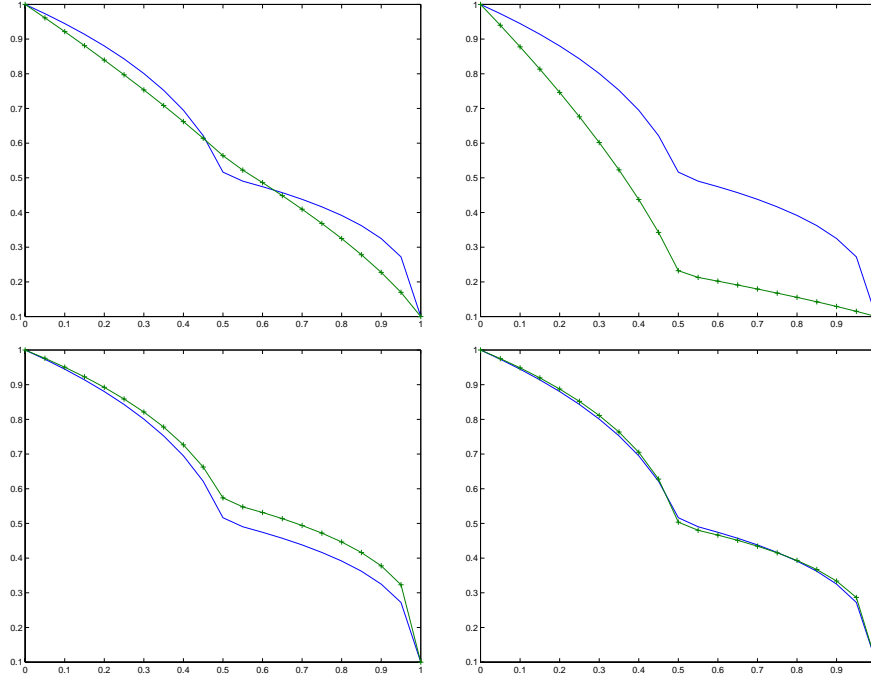


Figure 5: Initial and final temperature distributions for the radiation diffusion example with a starting point in each quadrant relative to the profile-matching parameters for $(\alpha, \beta_{right}) = (2.5, 10)$: upper left $(0.5, 1.5)$, upper right $(0.5, 12)$, lower right $(2.8, 8)$, lower right $(2.9, 15)$.

For the Hessian arithmetic complexity, we estimate the cost of applying each forward block to a vector. Assume that $h(x, u)$ and $\phi(x, u)$ are available and that all differentiated blocks are results of AD tools. Define C_ϕ , the cost of evaluating ϕ ; q , $1 + \text{number of nonzero rows in } \phi''$; and r , an implementation-dependent “constant,” typically ranging from 3 to 100. Then the cost of the Hessian-vector products can be estimated from the last two rows of Table 2.

For the inverse blocks, we need only low-quality approximations or limited-memory updates [9] of the square systems J_x^{-1} , J_x^{-T} , and H^{-1} .

The complexities for *all* operations required to apply the full-system matrix-vector product and its preconditioner are at worst linear in n or m , with coefficients that depend upon chromatic numbers (affected by stencil connectivity and intercomponent coupling of the PDE, and by separability structure of the objective function) and the implementation efficiency of AD tools.

Table 2: Complexity of formation of matrix objects or matrix-vector actions using forward or hybrid modes of modern automatic differentiation software. The asterisk signifies that the reverse mode consumes memory, in a carefully drawn time-space trade-off, so r is implementation dependent.

Object	Cost: Forward Mode	Cost: Fastest (Hybrid) Mode
J_x, J_x^T	$p_x C_h$	$p_x C_h$
$J_u v$	$2C_h$	$2C_h$
$J_u^T v$	$p_u C_h$	$r C_h^*$
$W_{xx} v, W_{ux}^T v$	$p_x C_h + q C_\phi$	$r(C_h + C_\phi)^*$
$W_{uu} v, W_{ux} v$	$p_u C_h + q C_\phi$	$r(C_h + C_\phi)^*$

6 Summary and Future Plans

As in domain decomposition algorithms for PDE analysis, partitioning in PDE-equality constrained optimization may be used to improve some combination of robustness, conditioning, and concurrency. Orders of magnitude of savings may be available by converging the state variables and the design variables within the same outer iterative process, rather than a conventional SQP process that exactly satisfies the auxiliary state constraints.

As with any Newton method, globalization strategies are important. These include parameter continuation (physical and algorithmic), mesh sequencing and multilevel iteration (for the PDE subsystem, at least; probably for controls, too), discretization order progression, and model fidelity progression. The KKT system appears to be a preconditioning challenge, but an exact factored preconditioner is known, and departures of preconditioned eigenvalues from unity can be quantified with comparisons of original blocks with blockwise substitutions in inexact models and solves. (For the full system, the KKT matrix will be nonnormal, so its spectrum does not tell all.)

With the extra, but automatable, work of forming Jacobian transposes and Hessian blocks, but no extra work in Jacobian preconditioning, any parallel analysis code may be converted into a parallel optimization code—and automatic differentiation tools will shortly make this relatively painless.

The gamut of PDE solvers based on partitioning should be mined for application to the KKT necessary conditions of constrained optimization and for direct use in inverting the state Jacobian blocks inside the optimizer.

We expect shortly to migrate our ADMAT/MATLAB code into the parallel ADIC/PETSc framework, while increasing physical dimensionality and parameter dimensionality. We will also tune the numerous preconditioning parameters for optimal parallel execution time. In the multidimensional large-scale context, we will incorporate multilevel Schwarz linear preconditioning for spatial Jacobian. Following the recent invention of the additive Schwarz preconditioned inexact Newton (ASPIN) [11], we will also experiment with full nonlinear preconditioning of the KKT system. This could include individual discipline optimizations as nonlinear preconditioner stages in a multidisciplinary computational optimization process—a key engineering (and software engineering) challenge of the coming years.

Acknowledgments

We acknowledge important discussions on algorithmics with George Biros (CMU), Xiao-Chuan Cai (UC-Boulder), Omar Ghattas (CMU), Xiaomin Jiao (UIUC), Tim Kelley (NCSU), Michael Wagner (ODU), and David Young (Boeing); on applications with Kyle Anderson (NASA Langley), Frank Graziani (LLNL), Dana Knoll (LANL), and Carol Woodward (LLNL); and on software with Satish Balay, Bill Gropp, Dinesh Kaushik, Barry Smith (all of Argonne) and Arun Verma (Cornell). This work was supported by the MICS subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract W-31-109-Eng-38; by Lawrence Livermore National Laboratory under ASCI Level-2 subcontract B347882 to Old Dominion University; by NASA under contract NAS1-19480 (while the second author was in residence at ICASE); and by the NSF under grant ECS-8957475. Computer facilities were provided by the DOE ASCI Alliance Program.

References

- [1] W. K. Anderson, W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. Achieving high sustained performance in an unstructured mesh CFD application. In *Proceedings of SC'99*. IEEE Computer Society, 1999. Gordon Bell Prize Award Paper in the “Special” Category.
- [2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. The Portable Extensible Toolkit for Scientific Computing (PETSc), version 28. <http://www.mcs.anl.gov/petsc/petsc.html>, 2000.
- [3] A. Battermann and M. Heinkenschloss. Preconditioners for Karush-Kuhn-Tucker matrices arising in optimal control of distributed systems. Technical Report TR96-34, Department of Computational and Applied Mathematics, Rice University, 1996.

- [4] G. Biros and O. Ghattas. Parallel Newton-Krylov methods for PDE-constrained optimization. In *Proceedings of SC99*. IEEE Computer Society, 1999.
- [5] G. Biros and O. Ghattas. A Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization. *SIAG/OPT Views-and-News*, 11(2):1–6, 2000.
- [6] C. Bischof, L. Roh, and A. Mauer. ADIC — An extensible automatic differentiation tool for ANSI-C. *Software-Practice and Experience*, 27(12):1427–1456, 1997.
- [7] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11:450–481, 1990.
- [8] P. N. Brown and C. S. Woodward. Preconditioning strategies for fully implicit radiation diffusion with material-energy transfer. Technical Report UCRL-JC-139087, Lawrence Livermore National Laboratory, 2000.
- [9] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited-memory methods. *Math. Prog., Ser. A*, 63:129–156, 1994.
- [10] X.-C. Cai. Some domain decomposition algorithms for nonselfadjoint elliptic and parabolic partial differential equations. Technical Report 461, Courant Institute, New York, 1989.
- [11] X.-C. Cai and D. E. Keyes. Nonlinearly preconditioned inexact Newton algorithms. <http://www.icas.edu/~keyes/papers/aspin.pdf>. Submitted to *SIAM J. Sci. Comput.*, 2000.
- [12] T. Coleman and A. Verma. ADMAT: An automatic differentiation toolbox for MATLAB. Technical report, Computer Science Department, Cornell University, 1998.
- [13] M. Dryja and O. B. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. Technical Report 339, Department of Computer Science, Courant Institute, 1987.
- [14] W. D. Gropp, D. K. Kaushik D. E. Keyes, and B. F. Smith. High performance parallel implicit CFD. To appear in *Parallel Computing*, 2000.
- [15] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM J. Num. Anal.*, 35:508–523, 1998.
- [16] D. E. Keyes and W. D. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM J. Sci. Stat. Comput.*, 8(2):s166–s202, 1987.
- [17] D. A. Knoll and W. J. Rider. A multigrid preconditioned Newton-Krylov method. *SIAM J. Sci. Stat. Comput.*, 21:691–710, 2000.
- [18] The MathWorks. Matlab 5.3.1. <http://www.mathworks.com/products/matlab>, 2000.
- [19] P. R. McHugh, D. A. Knoll, and D. E. Keyes. Application of a Newton-Krylov-Schwarz algorithm to low Mach number combustion. *AIAA J.*, 36:290–292, 1998.
- [20] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer Verlag, 1999.
- [21] A. Shestakov and J. Milovich. Applications of pseudo-transient continuation and Newton-Krylov methods to the Poisson-Boltzmann and radiation diffusion equations. Technical Report UCRL-JC-139339, Lawrence Livermore National Laboratory, 2000.
- [22] B. F. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Algorithms for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.